1 Creating a Simkit Simulation Model

After Simkit is installed and verified to be on the classpath, you can start writing Discrete Event Simulation (DES) models in Simkit. Simkit is designed to make it simple to create working simulation programs from an Event Graph model that has been constructed. Thus a Simkit DES program is simply the implementation of an Event Graph model. The model itself is an abstraction; there are many platforms on which an Event Graph model can be constructed. However, Simkit makes the process straightforward.

The first step, of course, is to have an Event Graph model developed. If the Event Graph model is relatively small (fewer than 20 events, say), then the entire model can be written as one class. For more complex models, a component design works best, however, and the model may be constructed using several classes and instances. Simkit provides the SimEventListener pattern to facilitate component design, but for now let's confine ourselves to a monolithic (single-class) approach.

Start by subclassing `simkit.SimEntityBase`, which provides the important methods that allow creation of the model. Each part of the Event Graph model may then be "mapped" into Simkit code by means of Table 1

2 Event Graph Model and Simkit

Table 1 below shows the relationship between basic elements of an Event Graph model and the corresponding Simkit implementation.

Event Graph	**Simkit**
Parameter	Private instance variable, setter and getter
State Variable	Protected instance variable, getter, no setter
Event	'do' method
Scheduling Edge	Call to `waitDelay()` in scheduling event's 'do' method
Run Event	`reset()` method to initialize state variables; `doRun()` method to fire PropertyChange events for time-varying state variables
Events scheduled from Run event	Calls to `waitDelay()` in `doRun()` method

Table 1. Relationship Between Event Graph and Simkit Model

The Simkit convention is that all state transitions are accompanied by firing a PropertyChangeEvent. If the state variable is a time-varying one (such as "number in queue"), then it should use the three-argument form for firePropertyChange(), where the arguments are name, old value, new value. Otherwise (for example, "time in the system"), the two-argument form should be used, where the arguments are property name, new value. Note that the name of the property is not necessarily the same as the state variable. Also, firing a PropertyChangeEvent *by itself* does not cause a state variable to change.